Trade-offs in High-Performance Numerical Library Design

Lois Curfman McInnes
Mathematics and Computer Science Division
Argonne National Laboratory

The Conference on High Speed Computing
April 22-25, 2002
Salishan Lodge, Gleneden Beach, Oregon

Outline

- Motivation
  - Complex, multi-physics, multi-scale applications
  - Distributed, multi-level memory hierarchies
- High-Performance Scientific Components
  - What are components?
  - Common Component Architecture (CCA)
  - Center for Component Technology for Terascale Simulation Software (CCTTSS)
- Parallel Components for PDEs and Optimization
  - Approach
  - Performance
- Ongoing Challenges

Collaborators

- Co-developers of PETSc
  - Satish Balay, Kris Buschelman, Bill Gropp, Dinesh Kaushik, Matt Knepley, Barry Smith, Hong Zhang
- Co-developers of TAO
  - Steve Benson, Jorge Moré, Jason Sarich
- CCA/CCTTSS collaborators
  - Include ANL, Indiana Univ., LANL, LLNL, ORNL, PNNL, SNL, Univ. of Utah, etc.
  - Led by Rob Armstrong (SNL)
  - Special thanks to L. Freitag and B. Norris

LANS

Acknowledgements

- U.S. Department of Energy – Office of Science
  - Core funding in the MCS Division of Argonne through the Mathematical, Information, and Computational Science (MICS) program
  - Advanced Computational Testing and Simulation (ACTS) toolkit
  - Scientific Discovery through Advanced Computing (SciDAC) program
- National Science Foundation
  - Multi-Model Multi-Domain Computational Methods in Aerodynamics and Acoustics

Motivating Scientific Applications

Physics

Optimization

Derivative Computation

Meshes

Discretization

Algebraic Solvers

Data Redistribution

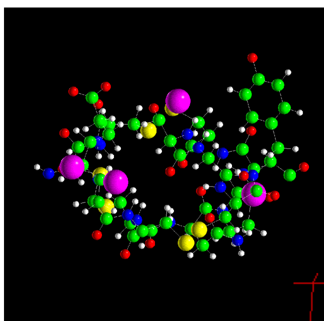
Parallel I/O

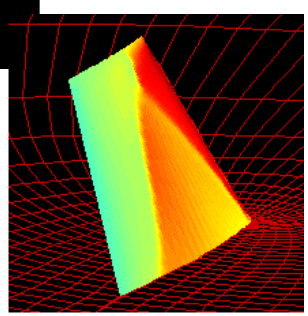
Adaptive Solution

Diagnostics

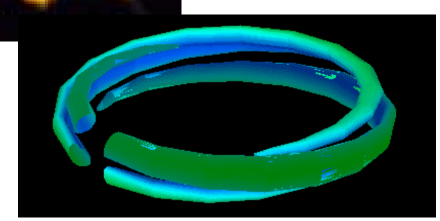
Steering

Visualization


Molecular structures

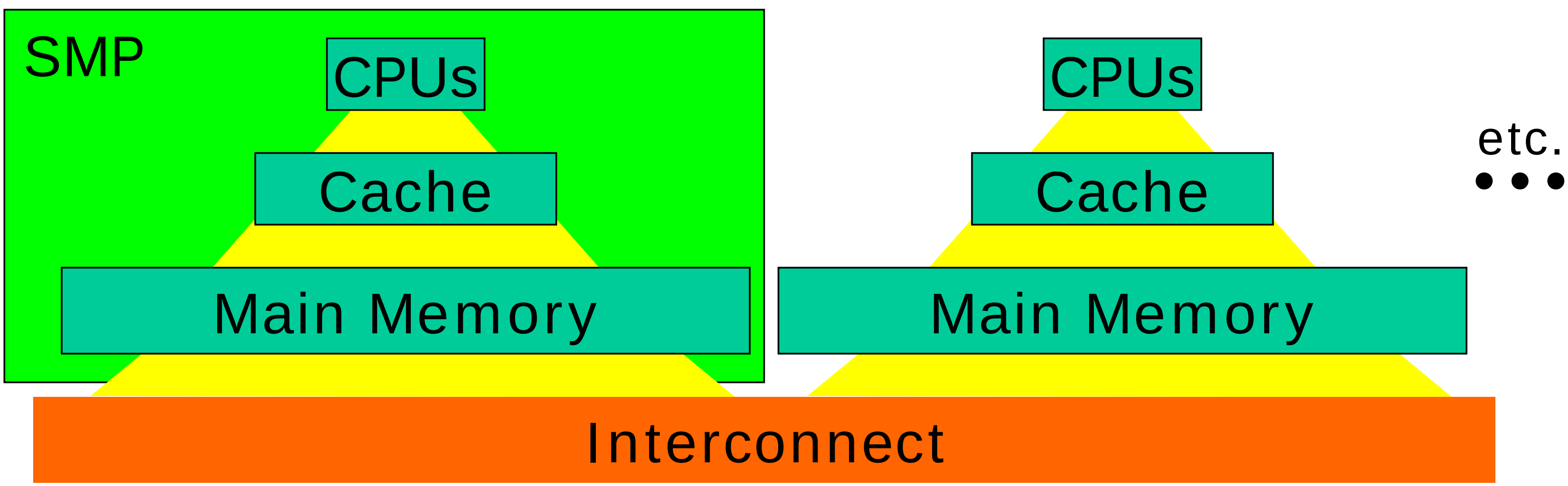

Aerodynamics


Astrophysics


Fusion

Target Architectures

- Systems have an increasingly deep memory hierarchy
- Time to reference main memory 100's of cycles



Challenges

- Community Perspective
  - Life-cycle costs of applications are increasing
    - Require the combined use of software developed by different groups
    - Difficult to leverage expert knowledge and advances in subfields
    - Difficult to obtain portable performance
- Individual Scientist Perspective
  - Too much energy focused on too many details
    - Little time to think about modeling, physics, mathematics
    - Fear of bad performance without custom code
    - Even when code reuse is possible, it is far too difficult
- Our Perspective
  - How to manage complexity?
    - Numerical software tools that work together
    - New algorithms (e.g., interactive/dynamic techniques, algorithm composition)
    - Multi-model, multi-physics simulations

Outline

- Motivation
  - Complex, multi-physics, multi-scale applications
  - Distributed, multi-level memory hierarchies
- **High-Performance Scientific Components**
  - What are components?
  - Common Component Architecture (CCA)
  - Center for Component Technology for Terascale Simulation Software (CCTTSS)
- Parallel Components for PDEs and Optimization
  - Approach
  - Performance
- Ongoing Challenges

Why Use Components?

Hero programmer producing single-purpose, monolithic, tightly-coupled parallel codes

- Promote software reuse
  - "The best software is code you don't have to write." *[Steve Jobs]*
- Reuse, through cost amortization, allows
  - thoroughly tested code
  - highly optimized code
  - developer team specialization
- Also reuse of skills, practice, and design

[Thanks to Craig Rasmussen (LANL) for the base of this slide.]

What are differences between objects and components?

- More similar than different
  - Object: a software black box
  - Component: object +
- OO techniques are useful for building individual components by relatively small teams; component technologies facilitate sharing of code developed by different groups by addressing issues in
  - Language interoperability
    - Via interface definition language (IDL)
  - Well-defined abstract interfaces
    - Enable “plug-and-play”
  - Dynamic composability
    - Components can discover information about their environment (e.g., interface discovery) from framework and connected components
- Can easily convert from an object orientation to a component orientation
  - Automatic tools can help with conversion (ongoing work by C. Rasmussen and M. Sottile, LANL)
- For more info: C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, ACM Press, New York, 1998

CCA History and Participants

- 1998: CCA Forum originated
  - Participation from researchers who were exploring one-to-one software interfacing in the DOE ACTS Toolkit program
  - Open to everyone interested in HPC components
  - See http://www.cca-forum.org
  - Active CCA Forum participants include
    - ANL - Lori Freitag, Kate Keahey, Jay Larson, Lois McInnes, Boyana Norris
    - Indiana Univ. - Randall Bramley, Dennis Gannon
    - LANL - Craig Rasmussen, Matt Sotille
    - LLNL - Scott Kohn, Gary Kumfert, Tom Epperly
    - ORNL - David Bernholdt, Jim Kohl
    - PNNL - Jarek Nieplocha, Theresa Windus
    - SNL - Rob Armstrong, Ben Allan, Curt Janssen, Jaideep Ray
    - Univ. of Utah - Steve Parker
    - And others as well …
- 2001: Center for Component Technology for Terascale Simulation Software (CCTTSS) founded
  - Support from the DOE SciDAC Initiative
  - CCTTSS team is a subset of the CCA Forum
  - Leader: Rob Armstrong (SNL)
  - See http://www.cca-forum.org/ccttss

CCTTSS Multi-Pronged Approach

CCTTSS Leader: Rob Armstrong (SNL)

- **HPC component specification and framework** – coordinator Scott Kohn (LLNL)
  - Unified reference framework implementation targeting both SPMD and distributed environments
  - Tools for language interoperability via a Scientific Interface Definition Language (SIDL)
- **Suite of scientific components** – coordinator Lois Curfman McInnes (ANL)
  - Linear and nonlinear algebra, optimization, mesh management, scientific data, visualization, steering, fault tolerance, scientific application domains, etc.
- **Parallel data redistribution** – coordinator Jim Kohl (ORNL)
  - Model coupling, visualization
- **Applications integration** – coordinator David Bernholdt (ORNL)
  - General outreach to the scientific community
  - Close feedback loop for users/developers of CCA technology
  - Collaborate with climate and chemistry applications domains as well as other groups

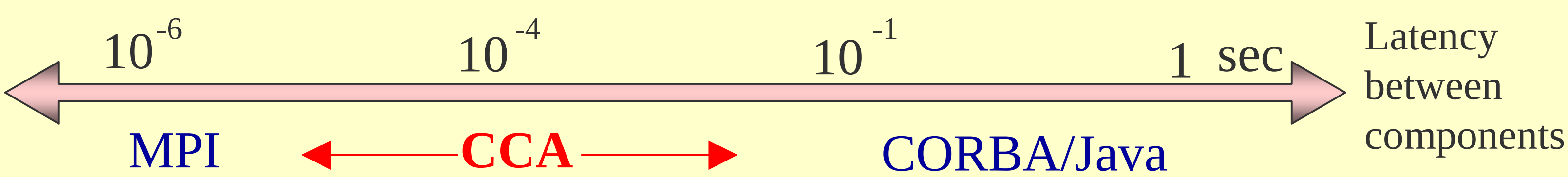
Requirements for a High-Performance Component Architecture

- Simple/Flexible
  - to adopt
  - to understand
  - to use
- Support a composition mechanism that does not impede high-performance component interactions
- Permit the SPMD paradigm in component form
- Meant to live with and rely on **other** commodity component frameworks to provide services …
  - e.g., JavaBeans, CORBA, ...

Goals of the Common Component Architecture (CCA)

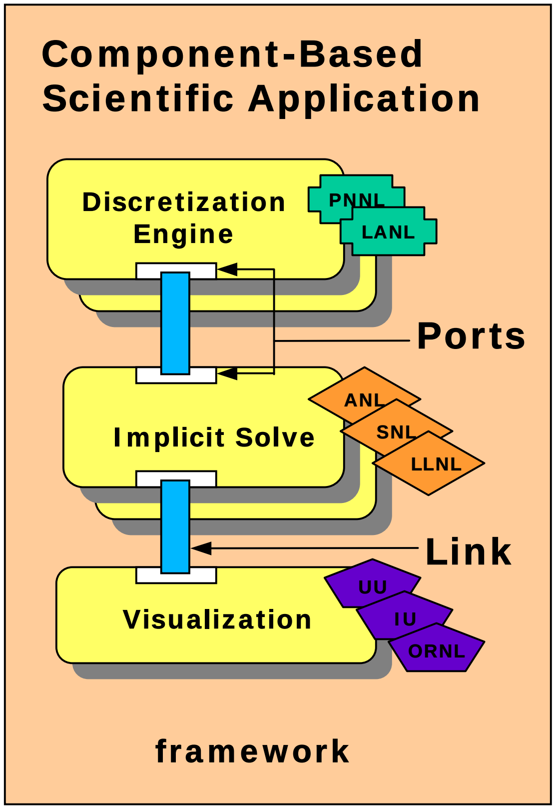
- Desire to build scientific applications by hooking together components
- DOE Common Component Architecture (CCA) provides a mechanism for interoperability of high-performance components developed by many different groups in different languages or frameworks.

Existing component architecture standards such as CORBA, Java Beans, and COM do not provide support for parallel components.



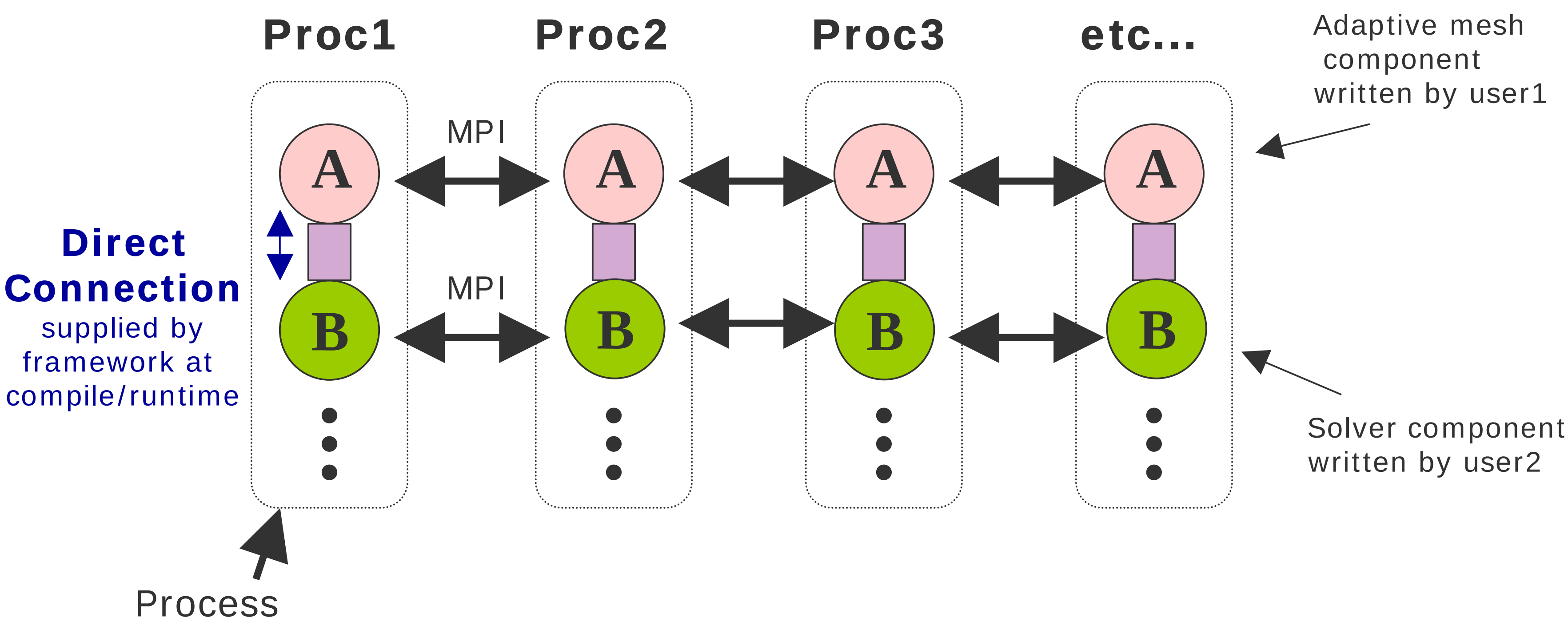
CCA Approach

- CCA specification dictates a basic set of interfaces (and corresponding behaviors) that components should implement to be CCA compliant.
  - *Ports* define the connection model for component interactions
  - *Provides/Uses* design pattern
- Components are manipulatable in a framework.
- CCA specification doesn't dictate frameworks or runtime environment.
  - Create components that are usable under a variety of frameworks
  - Provide a means for discovering interfaces
  - Specifically exclude *how* the components are linked; that is the job of a framework
  - Provide language-independent means for creating components (via SIDL)



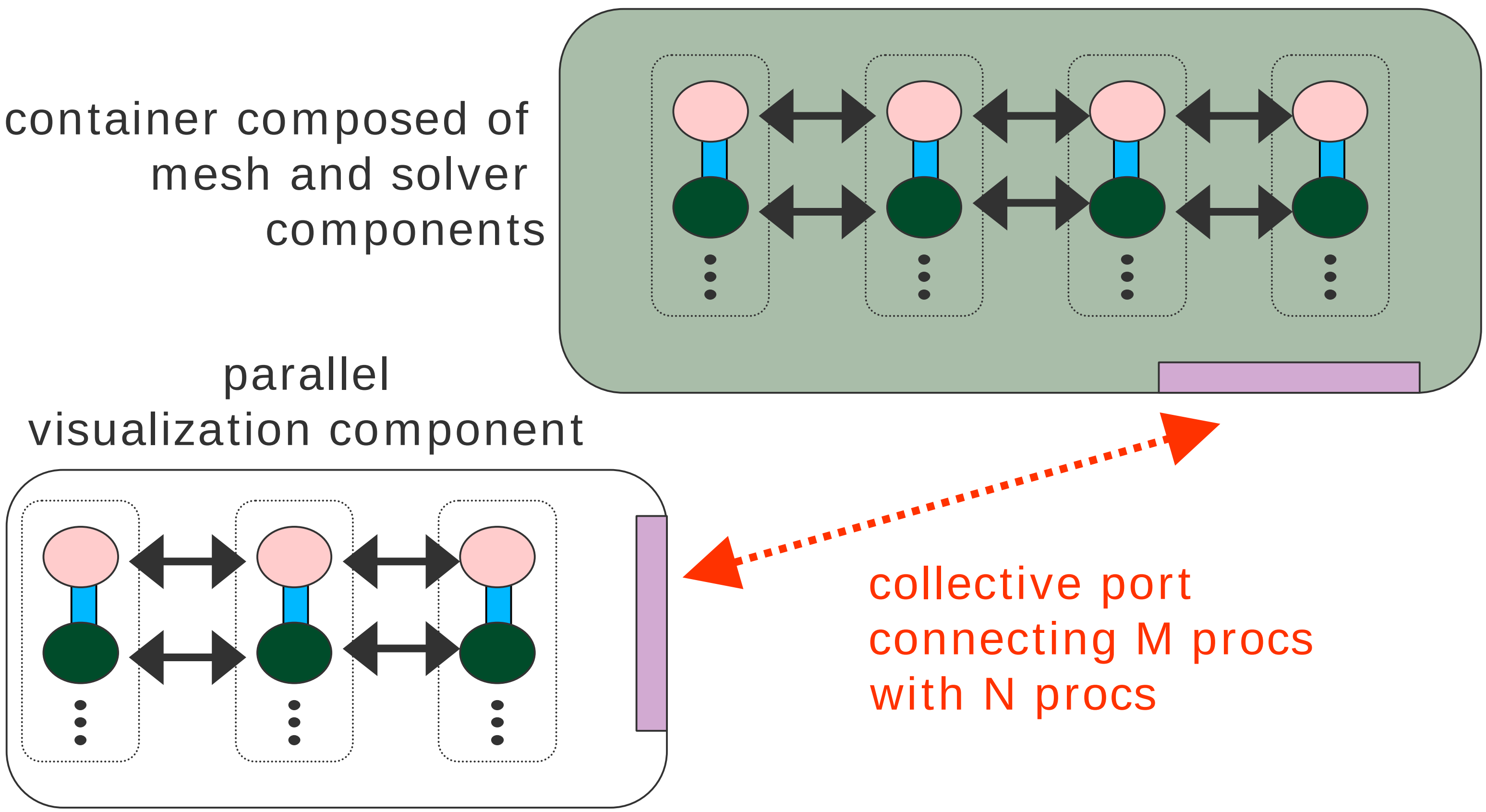
CCA Concept of SPMD Components

MPI application using CCA for interaction between components A and B within the same address space



CCA Collective Port Modularizes Processor/Data Decomposition

Combining previous parallel component with another parallel component in a different framework



CCA References

- Web sites
  - CCA Forum
    - http://www.cca-forum.org
  - Center for Component Technology for Terascale Simulation Software (CCA SciDAC Center)
    - http://www.cca-forum.org/ccttss
  - Sample component software and applications
    - http://www.cca-forum.org/cca-sc01
- Introductory paper
  - R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski, **Toward a Common Component Architecture for High-Performance Scientific Computing**, Proceedings of the High-Performance Distributed Computing Conference, pp. 115-124, 1999.

More CCA Papers

- B. Norris, S. Balay, S. Benson, L. Freitag, P. Hovland, L. McInnes, and B. Smith, **Parallel Components for PDEs and Optimization: Some Issues and Experiences**, preprint ANL/MCS-P932-0202, February 2002, Parallel Computing (to appear).
- B. Allan, R. Armstrong, A. Wolfe, J. Ray, D. Bernholdt, and J. Kohl, **The CCA Core Specification in a Distributed Memory SPMD Framework**, August 2001, Concurrency and Computation: Practice and Experience (to appear).
- T. Epperly, S. Kohn, and G. Kumfert. **Component Technology for High-Performance Scientific Simulation Software,** Proceedings of the International Federation for Information Processing's Working Conference on Software Architectures for Scientific Computing, 2000.
- S. Parker, **A Component-based Architecture for Parallel Multi-Physics PDE Simulations**, Proceedings of the 2002 International Conference on Computational Science (to appear).
- M. Sottile and C. Rasmussen, **Automated Component Creation for Legacy C++ and Fortran Codes**, Proceedings of the First International IFIP/ACM Working Conference on Component Deployment, June 2002 (submitted).
- R. Bramley, K. Chiu, S. Diwan, D. Gannon, M. Govindaraju, N. Mukhi, B. Temko, and M. Yechuri, **A Component Based Services Architecture for Building Distributed Applications,** Proceedings of High Performance Distributed Computing, 2000.
- K. Keahey, P. Beckman, and J. Ahrens, **Ligature: A Component Architecture for High-Performance Applications**, International Journal of High-Performance Computing Applications, 2000.

Related Work

- N. Furmento, A. Mayer, S. McGough, S. Newhouse, T. Field, and J. Darlington, **Optimization of Component-based Applications within a Grid Environment**, Proceedings of SC2001.
- C. René, T. Priol, and G. Alléon, **Code Coupling Using Parallel CORBA Objects**, Proceedings of the International Federation for Information Processing's Working Conference on Software Architectures for Scientific Computing, 2000.
- E. de Sturler, J. Hoeflinger, L. Kale, and M. Bhandarkar, **A New Approach to Software Integration Frameworks for Multi-physics Simulation Codes**, Proceedings of the International Federation for Information Processing's Working Conference on Software Architectures for Scientific Computing, 2000.
- R. Sistla, A. Dovi, P. Su, and R. Shanmugasundaram, **Aircraft Design Problem Implementation Under the Common Object Request Broker Architecture**, Proceedings of the 40th AIAA/ASME/ASCH/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 1999.

Outline

- Motivation
  - Complex, multi-physics, multi-scale applications
  - Distributed, multi-level memory hierarchies
- High-Performance Scientific Components
  - What are components?
  - Common Component Architecture (CCA)
  - Center for Component Technology for Terascale Simulation Software (CCTTSS)
- **Parallel Components for PDEs and Optimization**
  - Approach
  - Performance
- Ongoing Challenges

Software for Nonlinear PDEs and Related Optimization Problems

- **Goal**: For problems arising from PDEs, support the general solution of $F(u) = 0$

  User provides:
  - Code to evaluate $F(u)$
  - Code to evaluate Jacobian of $F(u)$ (optional)
    - or use sparse finite difference (FD) approximation
    - or use automatic differentiation (AD)
      - AD support via collaboration with P. Hovland and B. Norris (see http://www.mcs.anl.gov/autodiff)

- **Goal**: Solve related optimization problems, generally
  $min\ f(u),\ u_l < u < u_u\ ,\ c_l < c(u) < c_u$
  Simple example: unconstrained minimization: $min\ f(u)$

  User provides:
  - Code to evaluate $f(u)$
  - Code to evaluate gradient and Hessian of $f(u)$ (optional)
    - or use sparse FD or AD

What are the algorithmic needs of our target applications?

- Large-scale, PDE-based applications
  - multi-rate, multi-scale, multi-component
- Need
  - Fully or semi-implicit solvers
  - Multi-level algorithms
  - Support for adaptivity
  - Support for user-defined customizations (e.g., physics-informed preconditioners, transfer operators, and smoothers)

Reference: Salishan presentation by D. Keyes

Newton's Method

Nonlinear equations: Solve $f(u) = 0$, where $f: R^n \Rightarrow R^n$

$$f'(u^{l-1})\ \delta u^l = -f(u^{l-1})$$
$$u^l = u^{l-1} + \delta u^l$$

Solve approximately with preconditioned Krylov method

Unconstrained minimization: $\min f(u)$, where $f: R^n \Rightarrow R$

$$\nabla^2 f(u^{l-1})\ \delta u^l = -\nabla f(u^{l-1})$$
$$u^l = u^{l-1} + \delta u^l$$

Solve approximately with preconditioned Krylov method

- Can achieve quadratic convergence when sufficiently close to solution
- Can extend radius of convergence with line search strategies, trust region techniques, or pseudo-transient continuation.

Interface Issues

- How to hide complexity, yet allow customization and access to a range of algorithmic options?
- How to achieve portable performance?
- How to interface among external tools?
  - including multiple libraries developed by different groups that provide similar functionality (e.g., linear algebra software)
- Criteria for evaluation of success
  - efficiency (both per node performance and scalability)
  - usability
  - extensibility

Two-Phased Approach

- Phase 1
  - Develop parallel, object-oriented numerical libraries
    - OO techniques are quite effective for development with a moderate sized team
    - Provide foundation of algorithms, data structures, implementations
- Phase 2
  - Develop CCA-compliant component interfaces
    - Leverage existing code
    - Provide a more effective means for managing interactions among code developed by different groups

Parallel Numerical Libraries: PETSc and TAO

- ***PETSc: Portable, Extensible Toolkit for Scientific Computation***
  - S. Balay, K. Buschelman, B. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. Smith, H. Zhang
  - http://www.mcs.anl.gov/petsc
  - Targets the parallel solution of large-scale PDE-based applications
  - Begun in 1991, now over 8,500 downloads since 1995
- ***TAO: Toolkit for Advanced Optimization***
  - S. Benson, L. C. McInnes, J. Moré, J. Sarich
  - http://www.mcs.anl.gov/tao
  - Targets the solution of large-scale optimization problems
  - Begun in 1997 as part of DOE ACTS Toolkit
- Both are freely available and supported research toolkits
  - Hyperlinked documentation, many examples
  - Usable from Fortran 77/90, C, and C++
- Both are portable to any parallel system supporting MPI, including
  - Tightly coupled systems
    - Cray T3E, SGI Origin, IBM SP, HP 9000, Sun Enterprise
  - Loosely coupled systems, e.g., networks of workstations
    - Compaq, HP, IBM, SGI, Sun
    - PCs running Linux or Windows

Some Related Work in Numerical Libraries

(Not an exhaustive list)

- Krylov methods and preconditioners (for large, sparse problems)
  - Trilinos – Heroux et al. http://www.cs.sandia.gov/Trilinos
  - Parpre – Eijkhout and Chan http://www.cs.utk.edu/~eijkhout/parpre.html
  - Hypre – Cleary et al. http://www.llnl.gov/casc/hypre
  - SPARSKIT, etc. – Saad www.cs.umn.edu/~saad
- Nonlinear solvers
  - KINSOL – Hindmarsh http://www.llnl.gov/casc/PVODE
  - NITSol – Walker and Pernice
- Optimization software
  - Hilbert Class Library - Gockenback, Petro, and Symes http://www.trip.caam.rice.edu/txt/hcldoc/html
  - OPT++ - Meza http://csmr.ca.sandia.gov/projects/opt/opt++.html
  - DAKOTA - Eldred et al. http://endo.sandia.gov/DAKOTA
  - COOOL - Deng and Gouivera http://coool.mines.edu
  - Veltisto - Biros and Ghattas http://www.cs.nyu.edu/~biros/veltisto

Programming Model

- **Goals**
  - Portable, runs everywhere
  - Performance
  - Scalable parallelism
- **Approach**
  - Distributed memory, “shared-nothing”
    - Requires only a compiler (single node or processor)
    - Access to data on remote machines through MPI
  - Can still exploit “compiler discovered” parallelism on each node (e.g., SMP)
  - Hide within parallel objects the details of the communication
  - User orchestrates communication at a higher abstract level than message passing

PETSc Numerical Libraries

Nonlinear Solvers		
Newton-based Methods		Others
Line Search	Trust Region	

Time Steppers			
Euler	Backward Euler	Pseudo Time Stepping	Others

Krylov Subspace Methods							
GMRES	CG	CGS	Bi-CG-STAB	TFQMR	Richardson	Chebychev	Others

Preconditioners						
Additive Schwartz	Block Jacobi	Jacobi	ILU	ICC	LU (Sequential only)	Others

Matrices					
Compressed Sparse Row (AIJ)	Blocked Compressed Sparse Row (BAIJ)	Block Diagonal (BDIAG)	Dense	Matrix-free	Others

Distributed Arrays

Index Sets			
Indices	Block Indices	Stride	Others

Vectors

TAO Solvers

Unconstrained Minimization						
Newton-based Methods		Limited Memory Variable Metric (LMVM) Method	Conjugate Gradient Methods			Others
Line Search	Trust Region		Fletcher-Reeves	Polak-Ribiére	Polak-Ribiére-Plus	

Bound Constrained Optimization					
Newton Trust Region	GPCG	Interior Point	LMVM	KT	Others

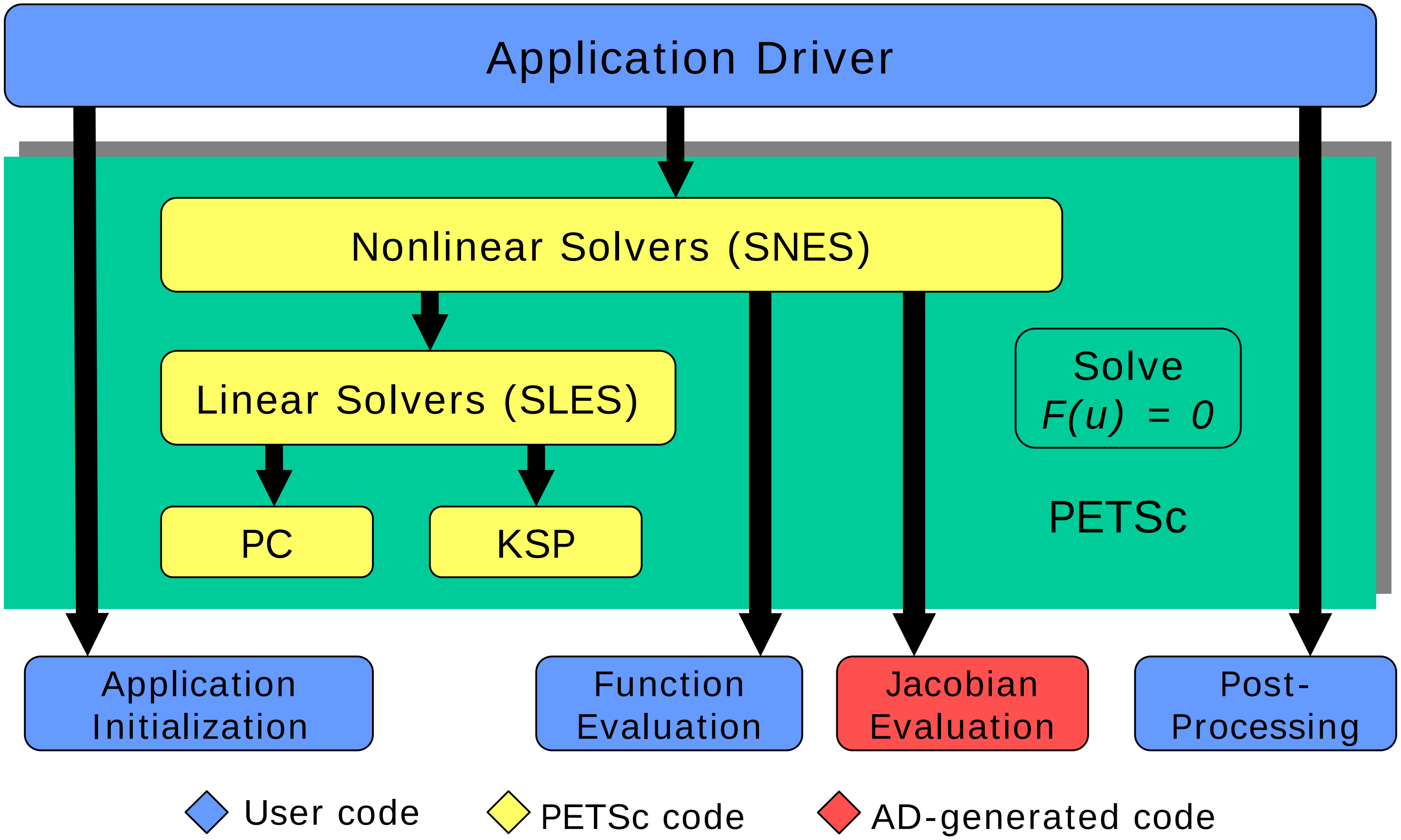
Nonlinear Least Squares					
Levenberg Marquardt	Gauss-Newton	LMVM	Levenberg Marquardt with Bound Constraints	LMVM with Bound Constraints	Others

Complementarity	
Semi-smooth Methods	Others

TAO interfaces to external libraries for parallel vectors, matrices, and linear solvers

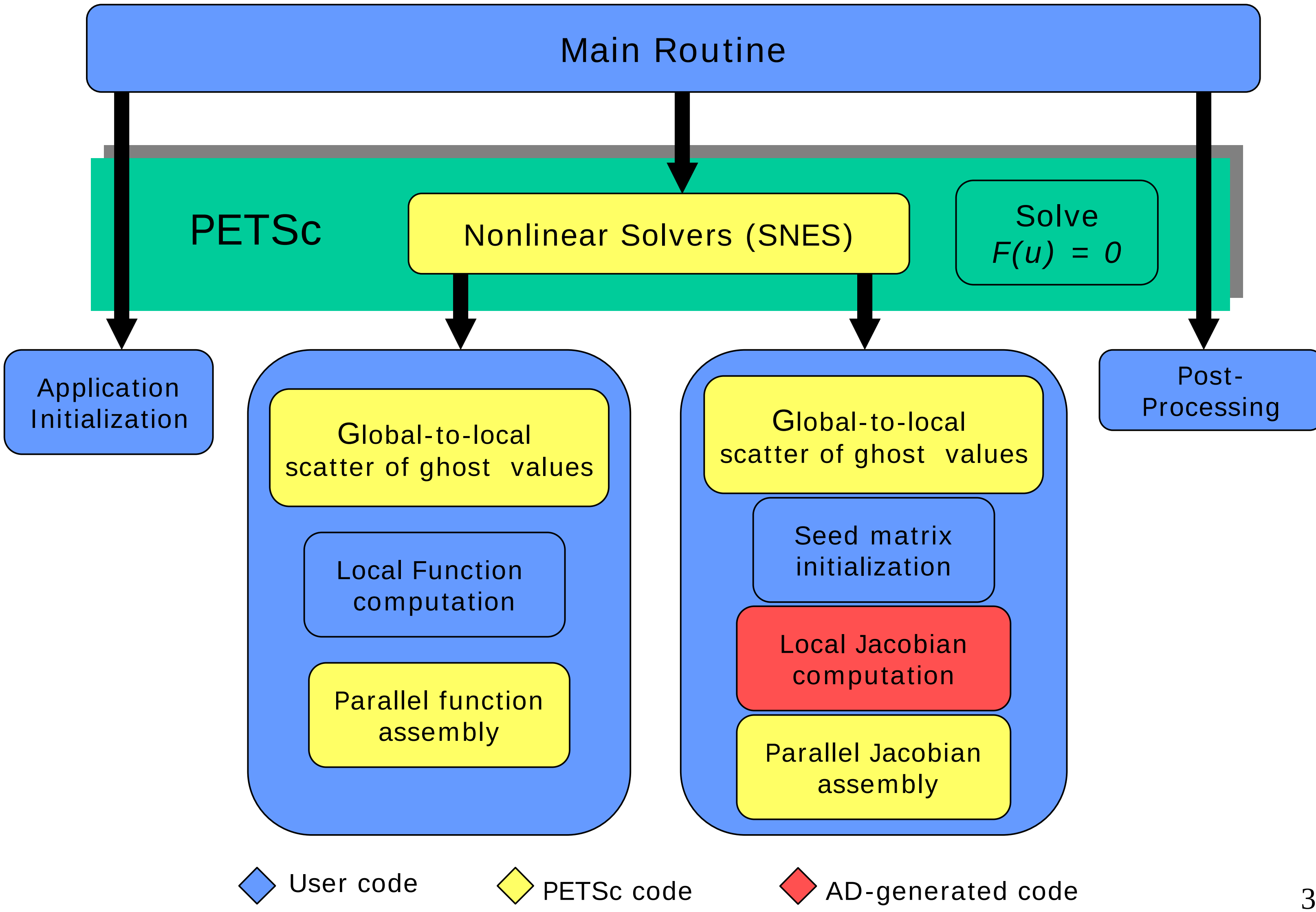
- **PETSc** (initial interface)
- **Trilinos** (SNL - new capability via ESI – thanks to M. Heroux and A. Williams)
- **Global Arrays** (PNNL – under development by J. Nieplocha et al.)
- Etc.

Nonlinear PDE Solution

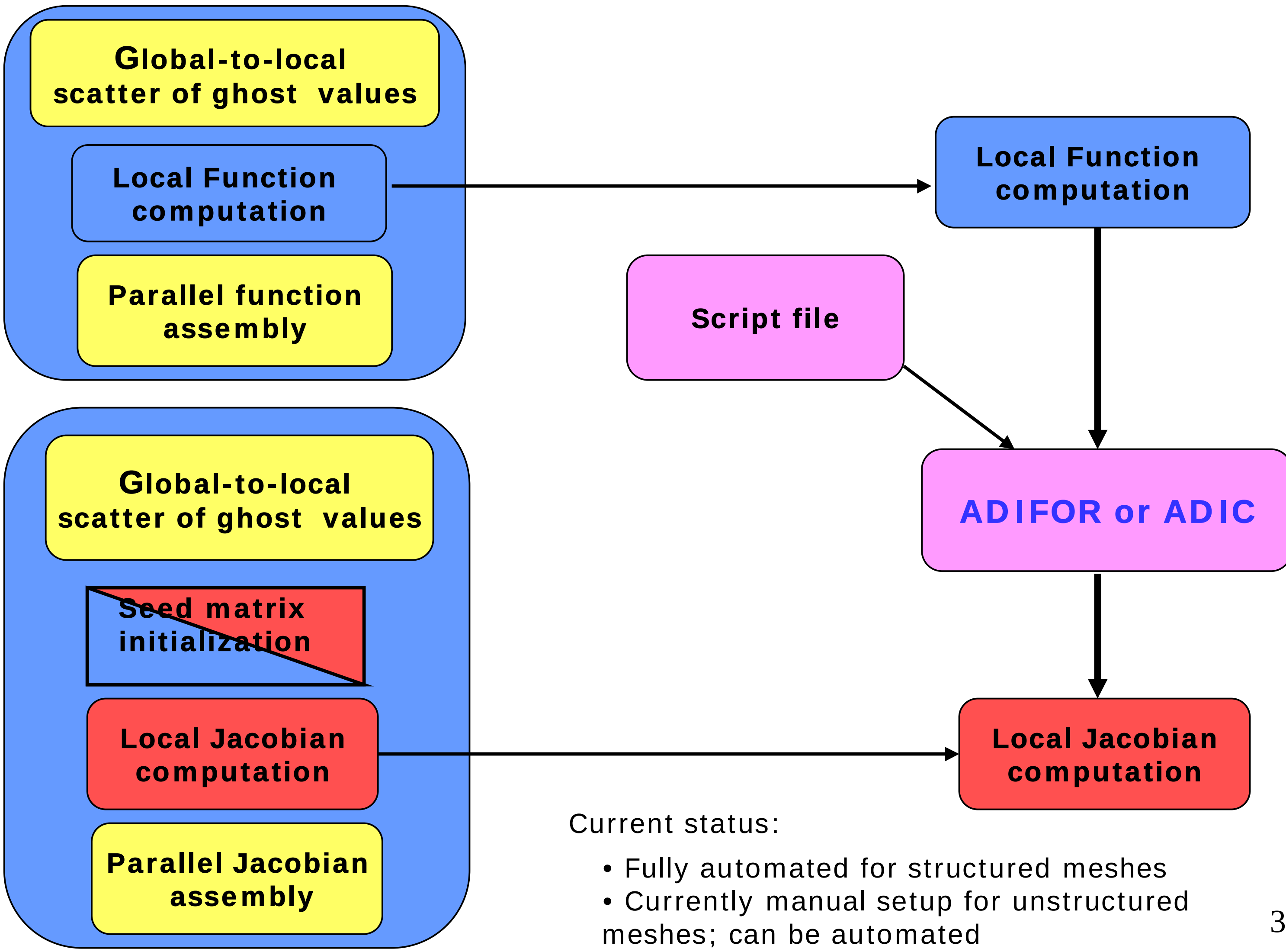


- **Automatic Differentiation (AD):** a technology for automatically augmenting computer programs, including arbitrarily complex simulations, with statements for the computation of derivatives, also known as sensitivities.
- **AD Collaborators**: P. Hovland and B. Norris (http://www.mcs.anl.gov/autodiff)

Nonlinear PDE Solution



Using AD with PETSc

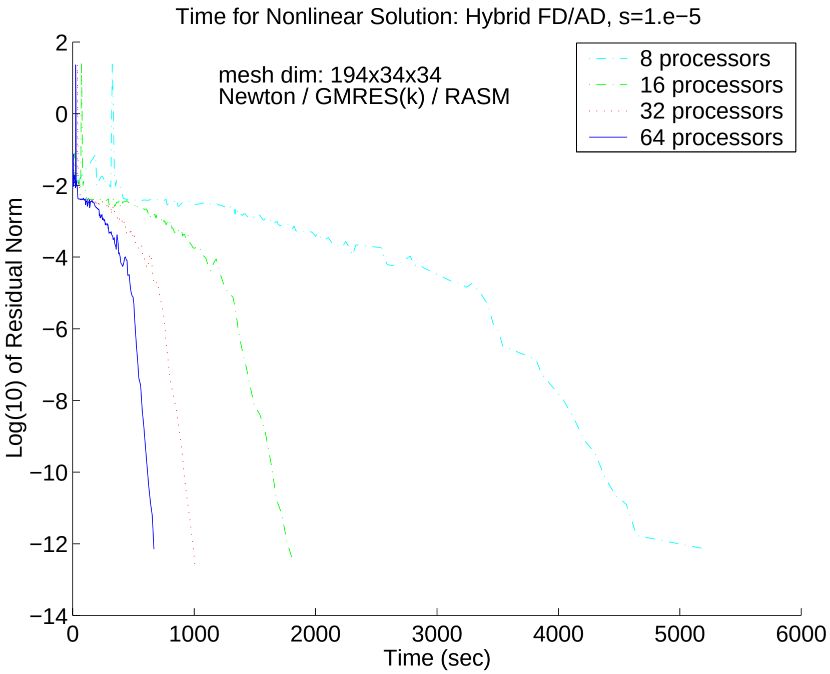


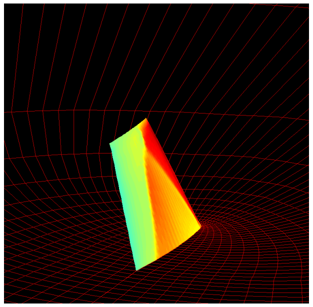
Current status:

- Fully automated for structured meshes
- Currently manual setup for unstructured meshes; can be automated

Hybrid FD/AD Strategy for Jacobian-vector Products

- FD
  - *$F'(x)\, v = [\, F(x+hv) - F(x)] / h$*
  - costs approximately 1 function evaluation
  - challenges in computing the differencing parameter, *h,* since we must balance truncation and round-off errors
- AD
  - costs approximately 2 function evaluations
  - no difficulties in parameter estimation
- Hybrid FD/AD
  - switch from FD to AD when $\|F\| / \|F_0\| < \delta$





Euler model; transonic flow over ONERA M6 wing

Some Experience in One-to-one Interfacing

Between PETSc and ...

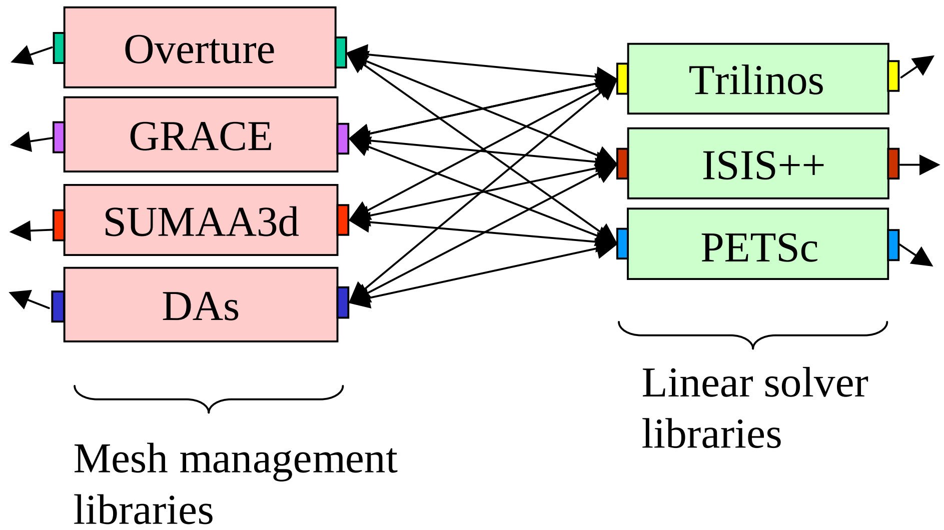
- Linear solvers
  - AMG http://www.mgnet.org/mgnet-codes-gmd.html
  - BlockSolve95 http://www.mcs.anl.gov/BlockSolve95
  - ILUTP http://www.cs.umn.edu/~saad/
  - LUSOL http://www.sbsi-sol-optimize.com
  - SPAI http://www.sam.math.ethz.ch/~grote/spai
  - SuperLU http://www.nersc.gov/~xiaoye/SuperLU
- Optimization software
  - TAO http://www.mcs.anl.gov/tao
  - Veltisto http://www.cs.nyu.edu/~biros/veltisto
- Mesh and discretization tools
  - Overture http://www.llnl.gov/CASC/Overture
  - SAMRAI http://www.llnl.gov/CASC/SAMRAI
  - SUMAA3d http://www.mcs.anl.gov/sumaa3d
- ODE solvers
  - PVODE http://www.llnl.gov/CASC/PVODE
- Others
  - Matlab http://www.mathworks.com
  - ParMETIS http://www.cs.umn.edu/~karypis/metis/parmetis

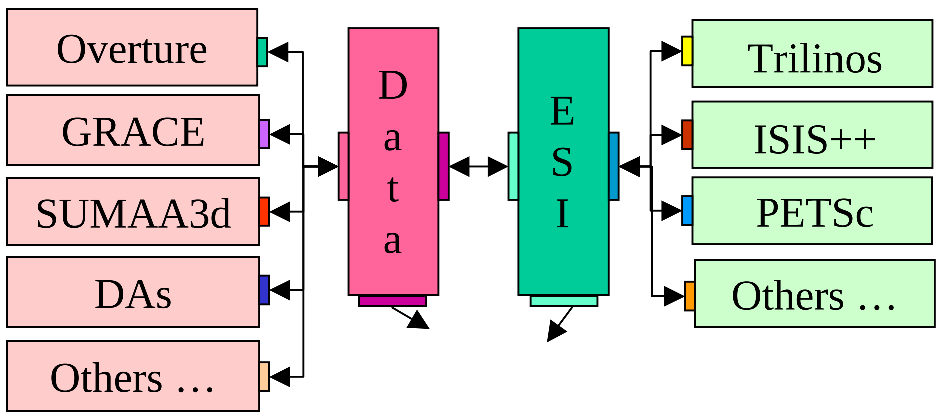
Between TAO and ...

- Linear solvers
  - PETSc http://www.mcs.anl.gov/petsc
- Optimization software
  - OOQP http://www.cs.wisc.edu/~swright/ooqp
  - APPSPACK http://cmsr.ca.sandia.gov/projects/apps.html

Common Interface Specification

- *Many-to-Many* couplings require *Many* ^{2} interfaces
  - Often a heroic effort to understand details of both codes
  - Not a scalable solution
- Common Interfaces: Reduce the *Many-to-Many* problem to a *Many-to-One* problem
  - Allow interchangeability and experimentation
  - Difficulties
    - Interface agreement
    - Functionality limitations
    - Maintaining performance





Current Interface Development Activities

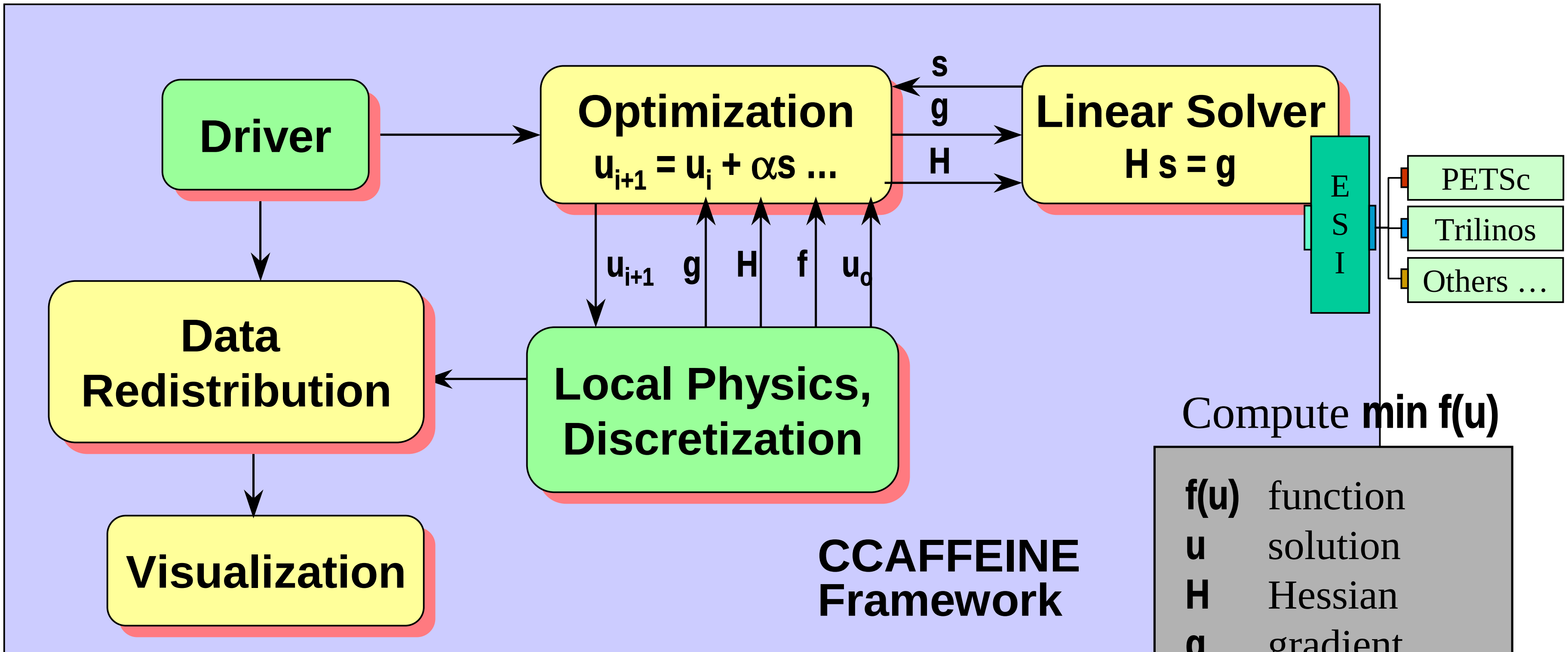
CCA Forum Scientific Data Components Working Group

- **Basic Scientific Data Objects**
  - Lead: David Bernholdt, ORNL
- **Unstructured Meshes**
  - Lead: Lori Freitag, ANL
  - in collaboration with TSTT (SciDAC ISIC)
- **Structured Adaptive Mesh Refinement**
  - Lead: Phil Colella, LBNL
  - in collaboration with APDEC (SciDAC ISIC)

Other Groups

- **Equation Solver Interface (ESI)**
  - Lead: Robert Clay (Terascale)
  - Predates CCA, but moving toward CCA compliance
- **MxN Parallel Data Redistribution**
  - Lead: Jim Kohl, ORNL
  - Part of CCTTSS MxN Thrust
- **Quantum Chemistry**
  - Leads: Curt Janssen, SNL; Theresa Windus, PNNL
  - Part of CCTTSS Applications Integration Thrust

Unconstrained Minimization Example Using CCA Components



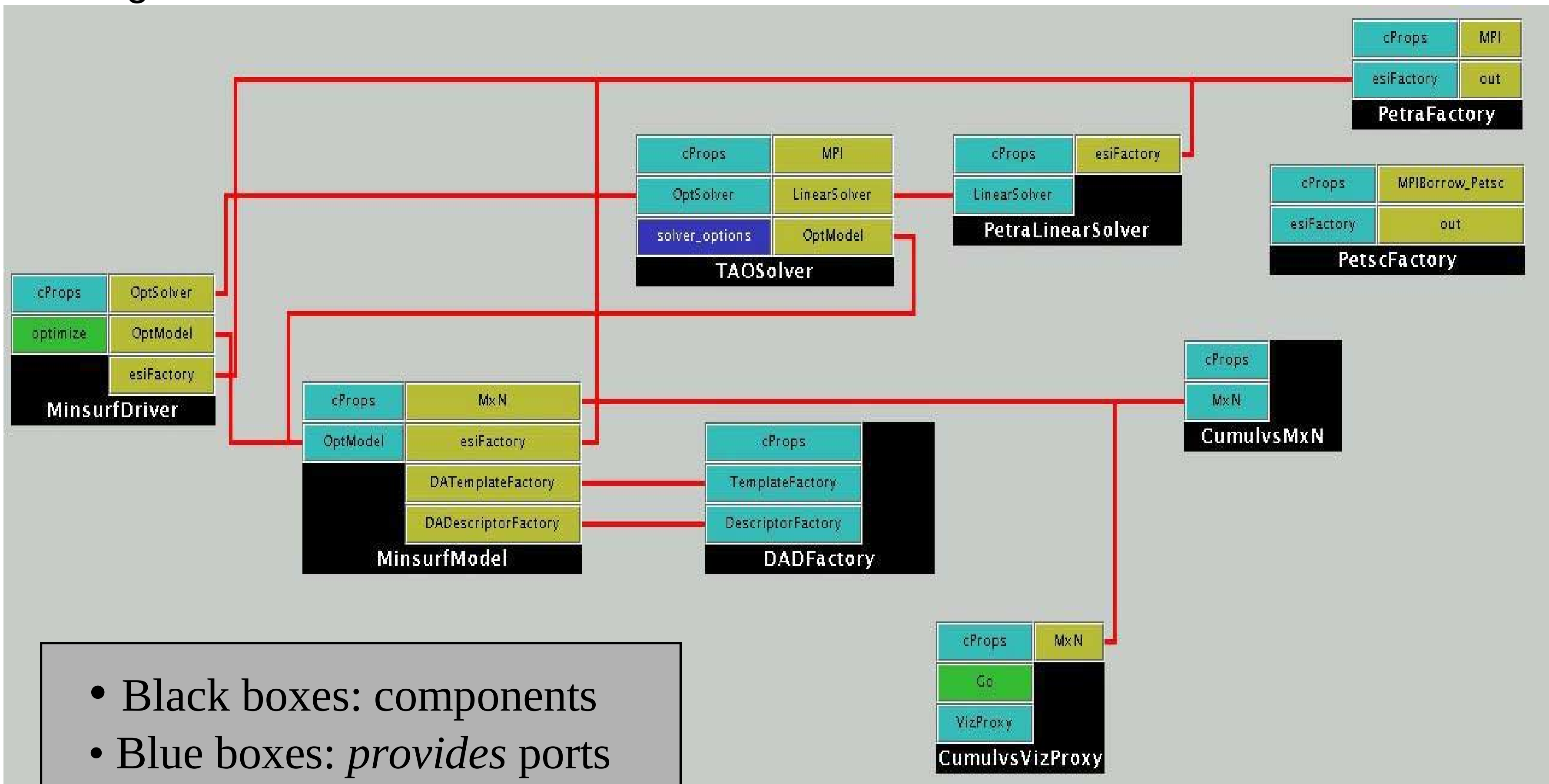
Compute **min f(u)**

f(u)	function
u	solution
H	Hessian
g	gradient
x	coordinates
c	connectivity
s	step direction

- CCAFFEINE – Common Component Architecture Fast Framework Example in Need of Everything
  - reference framework under development by B. Allan et al. (SNL)
  - http://www.cca-forum.org/cafe.html
- TAO – Toolkit for Advanced Optimization
  - http://www.mcs.anl.gov/tao
- Optimization component developers: S. Benson, L. C. McInnes, B. Norris, and J. Sarich

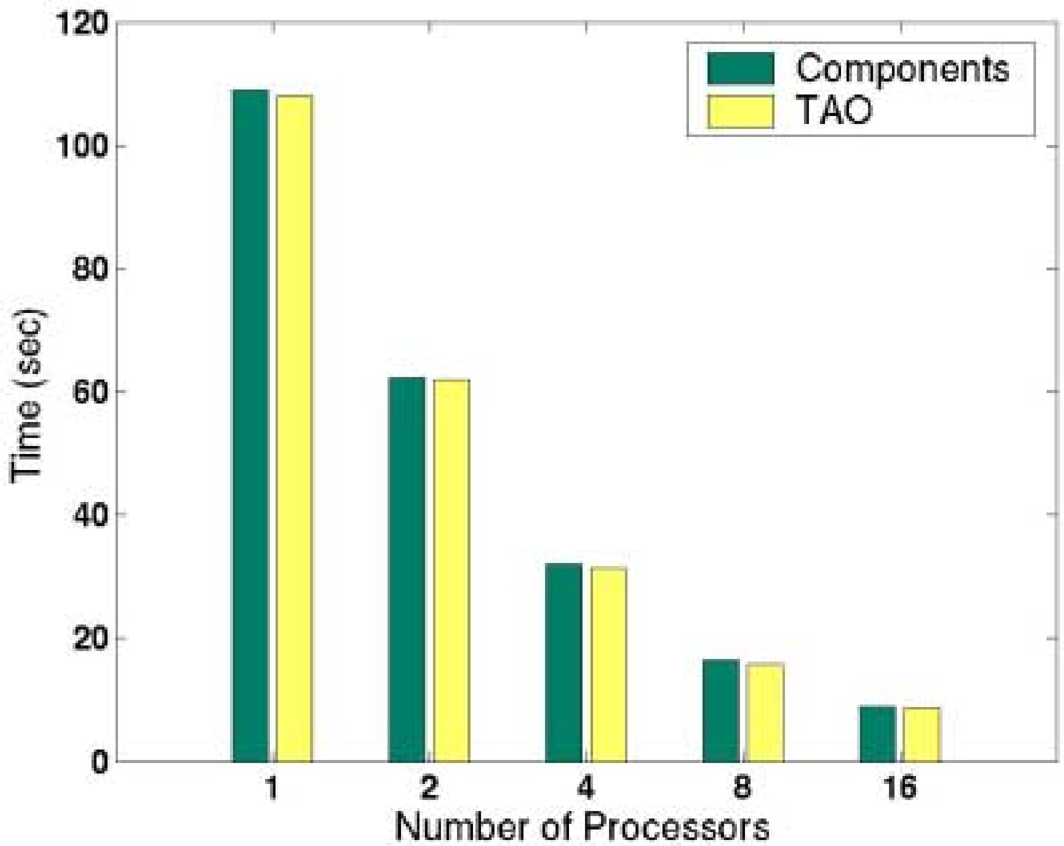
Component Wiring Diagram

Using GUI tool within CCAFFEINE framework

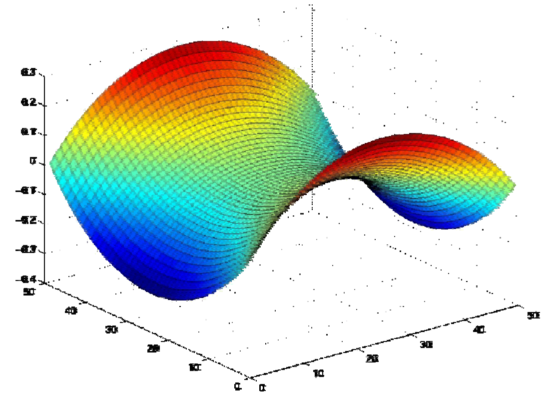


- Black boxes: components
- Blue boxes: *provides* ports
- Gold boxes: *uses* ports

Performance on a Linux Cluster



- Newton method with line search
- Solve linear systems with the conjugate gradient method and block Jacobi preconditioning (with no-fill incomplete factorization as each block's solver, and 1 block per process)
- Negligible component overhead; good scalability



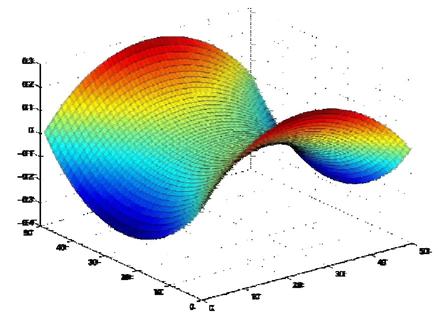
- Total execution time for a minimum surface minimization problem using a fixed-sized 250x250 mesh.
- Dual 550 MHz Pentium-III nodes with 1 G RAM each, connected via Myrinet

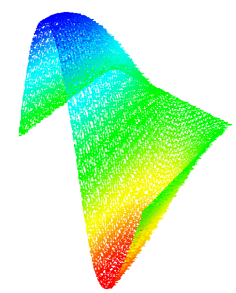
CCA Compliance in TAO

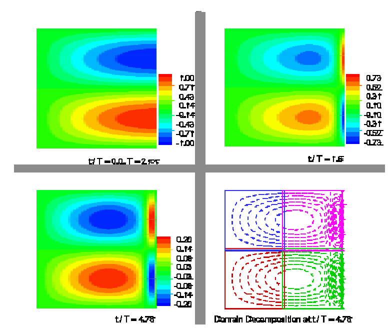
- Paradigm shift; both TAO and the application become components
  - Each is required to provide a default constructor and to implement the CCA component interface
    - contains one method: "setServices" to register ports
  - All interactions between components use ports
    - Application ***provides*** a "go" port and ***uses*** "taoSolver" port
    - TAO ***provides*** a "taoSolver" port
  - There is no "main" routine
- Status
  - TAO-1.4, released April 2002, includes CCA component interfaces
  - Ongoing work with T. Windus (PNNL) and C. Janssen (SNL) on CCA-based chemistry applications that involve optimization

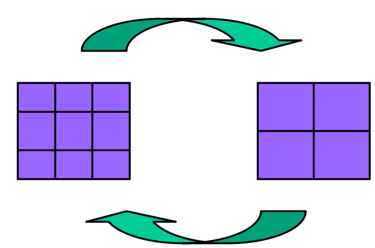
Sample CCA Components and Applications

- Developed by CCA working group for demonstration at SC01
- 4 applications using CCAFFEINE
  - Unconstrained minimization problem on a structured mesh
  - Time-dependent PDE on an unstructured mesh
  - Time-dependent PDE on an adaptive structured mesh
  - Ping-pong MxN
- More than 30 components
- Many components re-used in 3 apps
- Leverage and extend parallel software developed at different institutions
  - including CUMULVS, GrACE, MPICH, ODEPACK, PAWS, PETSc, PVM, TAO, and Trilinos
- Source code and documentation available via
  - http://www.cca-forum.org/cca-sc01









Component Re-Use

- Various services in CCAFFEINE
- Optimization solver
  - *TAOSolver*
- Integrator
  - *IntegratorLSODE*
- Linear solvers
  - *LinearSolver_Petra*
  - *LinearSolver_PETSc*
- Data description
  - *DADFactory*
- Data redistribution
  - *CumulvsMxN*
- Visualization
  - *CumulvsVizProxy*

Component interfaces to numerical libraries, all using ESI

Component interfaces to parallel data management and visualization tools

Summary

- Object-oriented techniques have been effective in enabling individual libraries for high-performance numerics to explore of trade-offs in
  - Algorithms, data structures, data distribution, etc.
- The CCA Forum is developing component technology specifically targeted at high-performance scientific simulations
  - Addressing issues in language interoperability, dynamic composability, abstract interfaces, parallel data redistribution, etc.
  - Aiming to enable the exploration of trade-offs in the broader context of multi-disciplinary simulations that require the combined use of software developed by different groups
- We have a solid start through an interdisciplinary, multi-institution team
  - Open to everyone interested in high-performance scientific components (see http://www.cca-forum.org for info on joining the CCA mailing list)
- **Lots** of research challenges remain!

One Challenge: Interfaces are central

- The CCA Forum participants do not pretend to be experts in all phases of computation, but rather just to be developing a standard way to exchange component capabilities.
- Medium of exchange: interfaces
  - **Need experts in various areas to define sets of domain-specific abstract interfaces**
    - scientific application domains, meshes, discretization, nonlinear solvers, optimization, visualization, etc.
- Developing common interfaces is difficult
  - Technical challenges
  - Social challenges

This means you!

Many, many additional research issues remain.

More Information

CCA: http://www.cca-forum.org
PETSc: http://www.mcs.anl.gov/petsc
TAO: http://www.mcs.anl.gov/tao